SANDIA REPORT
SAND2014-17970
Unlimited Release
Printed September 2014

Seismic Attenuation Inversion with t* Using tstarTomog

Leiph A. Preston

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd.
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2014-17970
Unlimited Release
Printed September 2014

Seismic Attenuation Inversion with t* Using tstarTomog

Leiph A. Preston
Department of Geophysics and Atmospheric Sciences
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-MS0750

Abstract

Seismic attenuation is defined as the loss of the seismic wave amplitude as the wave propagates excluding losses strictly due to geometric spreading. Information gleaned from seismic waves can be utilized to solve for the attenuation properties of the earth. One method of solving for earth attenuation properties is called t*. This report will start by introducing the basic theory behind t* and delve into inverse theory as it pertains to how the algorithm called tstarTomog inverts for attenuation properties using t* observations. This report also describes how to use the tstarTomog package to go from observed data to a 3-D model of attenuation structure in the earth.

ACKNOWLEDGMENTS

The Source Physics Experiments (SPE) would not have been possible without the support of many people from several organizations. The authors wish to express their gratitude to the National Nuclear Security Administration, Defense Nuclear Nonproliferation Research and Development (DNN R&D), and the SPE working group, a multi-institutional and interdisciplinary group of scientists and engineers. This work was done by Sandia National Laboratories under award number DE-AC52-06NA25946.

CONTENTS

FIGURES

NOMENCLATURE

DOE	Department of Energy
Q	Seismic quality factor (unitless)
SNL	Sandia National Laboratories
t*	T-star

1. INTRODUCTION

Seismic attenuation is defined as the loss of the seismic wave amplitude as the wave propagates excluding losses strictly due to geometric spreading. Attenuation mechanisms fall into two broad categories: intrinsic and scattering attenuation. Intrinsic attenuation is assumed to be micro-scale losses due to conversion of seismic wave energy into heat. Scattering attenuation is due to heterogeneities in the earth breaking up or scattering coherent single phases into multiple other phases. Thus, scattering is thought of as a seismic energy conserving process; the seismic energy is simply redistributed in space and time. In this brief, I will not distinguish between the two major attenuation mechanisms, only focusing on the losses due to attenuation from any mechanism.

2. ATTENUATION AND t* THEORY

2.1 t* Forward Model

Attenuation can be estimated in several different ways. In this paper, I will use the relatively simple measure called t*. All of the following derivations are based on equations from Lay and Wallace (1995). If we assume that a far-field seismic phase at a given distance from the source (r_0) has an amplitude of A_0, then in a simple homogeneous but attenuating medium, the amplitude, A, at distance r can be described by

$$A(x) = \frac{A_0}{r} \exp\left(-\frac{\pi f}{Qv} x \right) \tag{1}$$

where x is the measurement location, f is frequency (Hz), Q is defined as the seismic quality factor, and v is the wave speed. It will be noticed that x/v is the travel time. We can relax the homogeneous medium assumption by making v a function of position and r the length of the sinuous ray through a heterogenous medium. In a similar manner to how travel times are computed along 3-D ray paths, t* is defined as

$$t^* = \int \frac{dl}{Q(x)v(x)} = \int \frac{dt}{Q(x)} \tag{2}$$

integrated along the 3-D ray path. Here dl is a infinitesimal length along the ray path and dt is the travel time associated with dl. Of course, this can be discretized into a finite sum

$$t^* \approx \sum l_i s_i q_i \tag{3}$$

where l_i is the ray length for a given cell, i, s_i is the P- or S-wave slowness (inverse of velocity, $s_i = 1/v_i$) and q_i is the seismic loss factor (inverse of Q, $q_i = 1/Q_i$). Equation 3 is what is actually used in the forward calculation portion of the algorithm due to its linearity in parameters.

Using equation 2 and placing it in Equation 1 for a heterogeneous medium, one obtains

$$A(x) = \frac{A_0}{r} \exp\left(-\pi f t^*\right) \tag{4}$$

In practice, the amplitude ratio A/A_0 is typically measured at a specified frequency as spectral ratios. For now and for simplicity, let's assume that A is corrected for its distance r to obtain

$$A_c = A_0 \exp\left(-\pi f t^*\right) \tag{5}$$

Now rearranging Equation 5 and taking the natural logarithm of both sides, one gets

$$t^*_{obs} = -\frac{1}{\pi f} \ln\left(\frac{A_c}{A_0}\right) \tag{6}$$

This defines the observed t* value. One provides the algorithm with $\ln\left(A_c/A_0\right)$ and it internally computes Equation 6 and utilizes t* throughout as its observables.

2.2 t* Inverse Problem

Equation 3 defines the forward (predicted) calculation of t* based on a given model and Equation 6 that of the observed t*. We are interested in finding the Q values as a function of position that best describe the observed t* data. To accomplish this we need to find the optimal change in Q from the starting model in order to make the residual t* (observed minus predicted) zero or as close to zero as possible in an L2 norm sense. Partially differentiating Equation 3 with respect to q_i yields

$$\frac{\delta t^*}{\delta q_i} = l_i s_i \tag{7}$$

for a given ray path. Equation 7 is the sensitivity of changes in t* to changes in q_i. q_i is the loss factor associated with segment *i* of the ray path and the product $l_i s_i$ is simply the travel time of that segment, i.e., how long it takes a seismic wave to traverse the segment.

2.3 Differential t*

Equation 4 can be used directly if the source amplitude at a given distance is known. However, this often is not the case. More often, one is comparing two different receiver locations for a source of unknown amplitude. In this case, really what one is measuring is A_2/A_1 for locations 1 and 2. Taking ratios of Equation 4 for two different measurement locations 1 and 2 at the same frequency one obtains:

$$\frac{A_2}{A_1} = \frac{r_1 \exp\left(-\pi f t_2^*\right)}{r_2 \exp\left(-\pi f t_1^*\right)} \tag{8}$$

With some manipulation, it yields

$$\Delta t^* = t_2^* - t_1^* = -\frac{1}{\pi f}\ln\left(\frac{A_2}{A_1}\right) - \frac{1}{\pi f}\ln\left(\frac{r_2}{r_1}\right) \tag{9}$$

What one actually measures here is the first term on the right side of the equation. The second term on the right side is the correction term for location 1 and 2 being at different distances from the source. If they are at the same distance, this term vanishes. This latter term can also be ignored if the two amplitudes have already been corrected for distance. Equation 9 says that the amplitude ratio between two different receivers for the same source is just the difference between the two independent t* calculations. Thus, the predicted t* for this case is simply the difference between two calculations using Equation 3. The sensitivities are simply the difference in sensitivities computed by Equation 7.

2.4 Station and Source Corrections

Various emplacement conditions of sources can affect source output even with all other aspects of the source being equivalent. Similarly, receiver sensitivity to input signal can be affected by local emplacement conditions. In order to account for these unknown effects, we can augment our solution to find static source and/or receiver biases as well. This is accomplished by adding an unknown for each receiver and/or source. This unknown will absorb a portion of the total t* for each station and/or source in the model. These can be regularized like all other variables in the model. High regularization weights given to these parameters cause these corrections to be smaller, whereas small regularization weights allow them to approach the mean t* for that receiver or source.

3. THE DISCRETE FORWARD PROBLEM

3.1 Model Discretization

The model space is discretized into rectangular elements of equal sizes. The X dimension is divided into NX nodes with constant spacing dx. Similarly, the Y and Z dimensions are divided into NY and NZ nodes with constant spacings dy and dz, respectively. dx, dy and dz do not need to be co-equal.

Both a seismic velocity model and a Q model must be input and discretized onto the same grid. Internally, seismic velocity is converted to slowness, and Q is converted to the seismic loss factor, as stated in Section 2 in the explanation of Equation 3. This is for computational efficiency.

3.2 Ray Forward Problem

We will be solving the linear system of equations expressed by

$$\mathbf{Ax} = \mathbf{b} \tag{10}$$

where A is the matrix containing sensitivities of the data to changes in the model parameters; x is the column vector of unknown perturbations to the model parameters to be solved for; and b is the column vector of residuals. The the new model, at iteration *i+1,* will be

$$\mathbf{m}_{i+1} = \mathbf{m}_i + \mathbf{x}$$

We can apply the equations laid out in Section 2 to build up the A matrix. The A matrix contains the sensitivities of the forward problem to the model parameters (unknowns). The A matrix will have as many rows as there are data and as many columns as there are unknowns. Thus, if there are no receiver or source corrections and only P-wave t*, A will have NX*NY*NZ columns. If receiver and source corrections are also unknowns, then there will be NX*NY*NZ + nR + nS columns, where nR is the number of receivers and nS is the number of sources. For a given source-receiver pair, the seismic ray path in 3-D must be traced through a fixed seismic velocity model (Vidale and Hole, 1995; Zelt, 1990). This model does not change during the entire inversion process; thus, the ray paths remain the same across all iterations. Once the ray path is known, we can use Equation 3 to compute the predicted t* based on the present Q model. If receiver and source corrections are also part of the solution, then the ray-path-predicted t* is augmented with the current receiver or source correction.

To fill the A matrix we must first distribute the information from the 3-D ray path onto the discretized model grid. This is accomplished by dividing the ray path into segments each approximately 1/10 of the minimum node spacing and then distributing that segment's sensitivity (Equation 7) in a trilinear manner to that segment's surrounding 8 grid nodes. All the segments in a given ray path are summed; thus, grid nodes surrounding the ray path in 3-D space accumulate the sensitivities along the ray path, while grid nodes outside the nearest neighbors of the ray path have zero sensitivity. Therefore, one row of the A matrix will contain terms of the form of Equation 7 at columns corresponding to grid nodes that the ray path "hit" and zeros at the grid nodes not "hit". Additionally, if receiver and source corrections are also solved for then the row of the A matrix will also contain a 1.0 in the columns corresponding to this data point's receiver and source, with zeros in all other source and receiver columns. A is therefore sparse, being mostly zeros. The general form of A is then

$$A = \begin{bmatrix} \Lambda & D_R & D_S \end{bmatrix}$$

where Λ is the ray path portion of the matrix (NX*NY*NZ columns), D_R is the receiver correction portion (nR columns), and D_S is the source correction portion (nS columns). Corresponding to this is the b column vector, which would have an entry for a row *i* as:

$$b_i = \sum_{j=path} l_j s_j q_j + t^*_{R_i} + t^*_{S_i}$$

For differential t*, a row of A and b are built up by differencing two different ray paths which may or may not share common sources or receivers. For example, let's assume that the amplitude ratio is the ratio of the amplitudes of source 1S, receiver 1R to source 2S, receiver 2R. In this case, we can break up the problem into two parts that are individually identical to the standard (non-differential) case. First, a temporary A row, called A1, can be built as in the preceding paragraph using the ray from source 1S to 1R. Also, a temporary b (call it b1) can also be constructed as stated above for this ray. Similarly, another temporary A row, called A2, can be build from the ray 2S to 2R and another temporary b, called b2, can be computed. The final, differential A row will then be A1-A2 and final b will be b1-b2.

3.3 Regularization

Given the uneven distribution of ray paths from the data, some grid nodes will be hit multiple times by different data points, whereas others will not be hit by any data points at all. This will give rise to some portions of the model space being overdetermined while other parts are underdetermined. One method of handling this problem is by using regularization. Regularization serves to both bring some determinism to those nodes unhit by data and to control the solution such that poor or bad data do not overly influence the solution. The regularization matrix is called L. L will have the same number of columns as A, but the number of rows will be NX*NY*NZ + nR + nS, if P-wave t* is used with receiver and source correction terms. Regularization of the model portion of L is accomplished by imposing ansiotropic Laplacian smoothing to the Q model. The second order accurate discrete Laplacian operator is applied to the model, with a different, user-controllable weight being given to the vertical part of the operator compared to the horizontal portions. Each row corresponds to a different point in the 3-D grid acting as the pivot point for the Lapacian operator and will contain 7 entries. If the pivot point in the grid is at point x, y, z, then the entires will be:

$$L_{x+1,y,z} = L_{x-1,y,z} = L_{x,y+1,z} = L_{x,y-1,z} = 1$$
$$L_{x,y,z+1} = L_{x,y,z-1} = \zeta$$
$$L_{x,y,z} = -4 - 2\zeta$$

where ζ is the anisotropic vertical smoothing parameter. The corresponding entry to the regularization portion of the b column vector, b_L, is

$$b_{LSm} = -L_{Sm} m$$

where L_{Sm} is the Laplacian portion of the regularization matrix and m is the current Q model. This enforces the idea that we want a smooth Q model, not smooth perturbations to the Q model.

If one notices, the above equation uses the Laplacian of the model, which, in actuality, is 1/Q (the loss factor), not Q itself. Perhaps, however, one does not want the model to be smooth in 1/Q, but smooth in Q, or perhaps even one would want smoothness in terms of relative changes in Q (i.e., $\Delta Q / Q$). When 1/Q is the basis for smoothness, then high Q regions can appear very

rough, while low Q regions are much smoother; when Q is the basis of smoothness, then just the opposite situation would be seen; when relative Q is the basis of smoothness, then both low and high Q regions will have approximately the same apparent roughness. The default is smoothness based on 1/Q. To use Q as the basis, then the Laplacian portion of the regularization matrix must be scaled by Q^2; which means that each row is scaled by the square of the current value of the Q model for that row's pivot point. To use relative Q as a basis for smoothness, the Laplacian must be scaled by Q, making the Laplacian a relative scale independent of the size of Q at each point.

The receiver and source portions of the regularization matrix are simple. These are simply the identity matrix within their respective subsections of L. The b_L for these portions are the negative of their current values: we desire these parameters to be small, not just the perturbations.

The Laplacian, receiver correction, and source correction portions of the regularization matrix can be weighted independently from each other. These weights control how strongly the regularization affects the solution. A larger regularization weight on the smoothing portion will make for a smoother model that doesn't fit the data as well, in general, than a lower weight. Thus, there is an art and science to determining the proper regularization weights to apply. This is somewhat subjective, based on the data quality in conjunction with modeler expectations and biases.

3.3.1 Model Constraints

A further way to control the model is via model parameter constraints, which will attempt to keep model Q values close to the provided values. These can either be from previously known or estimated Q values or simply based on *a priori* beliefs. These *a priori* Q model values can be included in the inversion with various weights which control how closely the inversion will match these input values. High weights will force the model to strictly match the values, whereas lower weights will allow deviations from these values. Note that these weights are independent of the overall regularization weight, λ.

3.3.2 Final Regularization

Combining the above regularization matrices into a final form, L, gives

$$L = \begin{bmatrix} L_{Sm} & 0 & 0 \\ 0 & \alpha H_R & 0 \\ 0 & 0 & \beta H_S \\ \gamma / \lambda C & 0 & 0 \end{bmatrix}$$

where L_{Sm} is the Laplacian smoothing portion of the regularization, H_R is the receiver correction portion, H_S is the source correction portion, and C is the model constraint portion. The three columns shown in L are, from left to right, the NX*NY*NZ columns of the Q model,

the nR columns of the receiver correction terms, and the nS source correction columns. The scalars α and β are the independent weights given to the receiver and source correction portions of the regularization. γ is a diagonal matrix with entries giving the weights of the corresponding constraint values, and λ is the overall regularization weight (discussed in Section 4). The full L and b_L equations are then

$$\begin{bmatrix} L_{Sm} & 0 & 0 \\ 0 & \alpha H_R & 0 \\ 0 & 0 & \beta H_S \\ \gamma/\lambda C & 0 & 0 \end{bmatrix} x = \begin{bmatrix} -L_{Sm} m_i \\ -\alpha h_R \\ -\beta h_S \\ \gamma/\lambda (m_i - m_C) \end{bmatrix} \tag{11}$$

where m_i is the current model and m_C are *a priori* values for the model and h_R and h_S are vectors containing the current values of the receiver and source correction values, respectively.

3.3.3 Bounding Model Parameters

Another means of controlling the solution is via model parameter constraints. Maximum and minimum allowable values for Q can be imposed as a function of depth, if desired. These constraints are directly incorporated into the matrix inversion step (Bierlaire, 1991) so that a self-consistent solution results, bounded by these constraints.

3.4 Qs Modeling

All the previous discussion assumed only P-wave t* was being utilized. However, S-wave t* can also be used. A separate 3-D grid of Qs is also constructed in this case. A seismic shear wave velocity model must be included when Qs is desired. Exactly the same procedures are utilized for S-wave t* as for P-wave t*. Separate receiver and source corrections for S-wave t* will also be made if requested. The number of columns is doubled when using both P- and S-wave t*. In addition separate regularization is used.

4. INVERSION

The system in Equation 10 is constructed by augmenting the data A and b with the regularization L and b_L (Equation 11) to give a final system given by,

$$\begin{bmatrix} wA \\ \lambda L \end{bmatrix} x = \begin{bmatrix} wb \\ \lambda b_L \end{bmatrix}$$

where w is a diagonal matrix with entries giving the weights of each of the data points. These weights would typically be reciprocals of estimated standard error estimates. λ is the overall regularization weight given to the entire L matrix. This system is passed to the CGLS (Paige and Saunders, 1982) algorithm to determine x.

Since the velocity model does not change, the ray paths do not change, and the problem is completely linear. Thus, it should converge in one iteration. If a second iteration were attempted, then the perturbations should theoretically be zero. However, a couple of things can alter this simple picture. Although numerical errors will cause non-zero second iteration perturbations, based on experience, these perturbations are quite small relative to the model size. True non-linearity, however, results if anything but the default smoothness based on 1/Q is used. This is because now the value of the Laplacian regularization portion of L is based on the Q-model, instead of being independent of it.

Because of this possible non-linearity, the solution vector, x, output from the CGLS algorithm may not actually be the "best" solution. In fact, it could actually be a worse fit than the prior iteration's model. To explore this possibility we assume that the direction of x is correct, but that the magnitude of x may be too large. Thus, the code progressively halves the solution magnitude until the minimum variance is found. The solution vector at this minimum variance is the one used to update the prior model.

4.1 Convergence

For a non-linear model, some set of criteria must be met before one says that the solution has been obtained to sufficient accuracy. Several different criteria are available to the user to control this exit condition. Convergence criteria include maximum absolute change in the model, maximum relative change in the model, maximum L2 norm size for the model perturbation, and maximum change in total model variance between iterations. All criteria must be met for convergence. A maximum number of iterations can also be set that will stop the inversion procedure irregardless of the other criteria. See the Appendix for flags that control convergence (-s family of flags)

5. BUILDING THE STARTING MODEL

5.1 Software Environment and Setup

The tstarTomog package is a collection of C and C++ source files and includes two main programs: makeModel and tstarTomog. As such, to compile these programs a compiler capable of compiling these languages with openMP support is required. A netcdf 3+ library must also be on your library path.

If you have executables of makeModel and tstarTomog, then the netcdf requirement depends on whether the netcdf library was statically linked internally to the executable. If it was bundled as a static internal library, then the netcdf library need not be on the machine. In all cases, it is required that standard C, C++, and openMP libraries reside on your runtime library path.

The next four sections will describe the basic elements required by makeModel to run as well as some of the most common command line argument flags used to control its operation. Figure 1 demonstrates the overall flow of raw data and inputs into makeModel and tstarTomog through output.

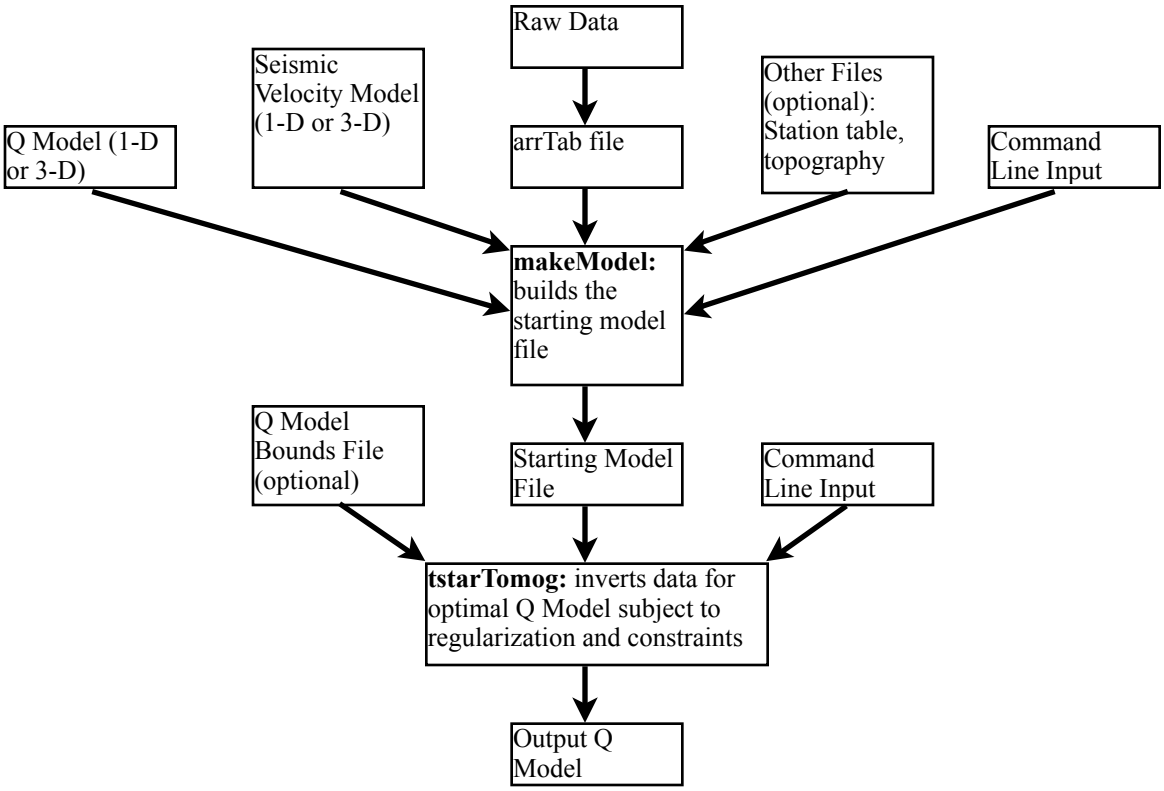

Figure 1: Overview chart showing flow of input, algorithms and output for the tstarTomog suite, which includes the bold face algorithms makeModel and tstarTomog.

5.2 Preparation for makeModel

As mentioned above the actual input data are $\ln\left(A_c / A_0\right)$. The first step is to build an 'arrTab' file.

This file is a flat ASCII text file with each line representing one data point. Each line contains all relevant information concerning that data point including source and receiver information, the data value itself, estimated error in the data value and data type. There are two varieties of arrTab files: one for absolute t* and another for differential t*. The absolute arrTab is completely self contained in that it does not require other files to complete its definition. The differential arrTab file requires a separate station table file to complete its definition.

5.2.1 Absolute arrTab Format

The format for an absolute t* arrTab file is:

```
evNa evLat evLon evDep stNa stLat stLon stEl freq data edata type
```

where,

evNa: unique event identifier name

evLat: event latitude
evLon: event longitude
evDep: depth (km) of the event relative to sea level. Note that this will be negative for events above sea level.
stNa: unique station identifier name (limited to 10 characters)
stLat: station latitude
stLon: station longitude
stEl: station elevation (km) relative to sea level. Note this has the opposite sign compared to evDep.
freq: frequency (Hz) where the data is measured
data: $\ln\left(A/A_0\right)$ at the specified frequency. A can either be corrected or uncorrected for distance.
edata: estimated error in the data in data units
type: the type of the data. This can either be 'P' for P-wave t* or 'S' for S-wave t*

Note that flags given to makeModel can change the interpretation of those fields with Lat, Lon, and Dep. See below for detail.

5.2.2 Differential arrTab Format

The format for a differential t* arrTab file is:

```
st1 st2 ev1 ev1Lat ev1Lon ev1Dep  ev2 ev2Lat ev2Lon ev2Dep freq data
edata type
```

where,

st1: unique station identifier name (limited to 10 characters) for the reference data
st2: unique station identifier name (limited to 10 characters) for the second data
ev1: unique event identifier name for the reference data
ev1Lat: event 1 latitude
ev1Lon: event 1 longitude
ev1Dep: event 1 depth (km) relative to sea level. Note that this will be negative for events above sea level.
ev2: unique event identifier name for the second data
ev2Lat: event 2 latitude
ev2Lon: event 2 longitude
ev2Dep: event 2 depth (km) relative to sea level. Note that this will be negative for events above sea level.
freq: frequency (Hz) where the data is measured
data: $\ln\left(A_2/A_1\right)$ at the specified frequency for amplitudes of event 2, station 2 divided by amplitudes of event 1, station 1.
edata: estimated error in the data in data units

type: the type of data. This can either be 'P' for P-wave t* or 'S' for S-wave t*

Note that flags given to makeModel can change the interpretation of those fields with Lat, Lon, and Dep. See below for detail.

5.2.3 Station Table Format

When a differential arrTab file is given, a station table file must also be provided. This is a flat ASCII text file with a line for each station. It has the following format:

```
stName stLat stLon stElev
```

where,

stName: unique station identifier name (limited to 10 characters)
stLat: station latitude
stLon: station longitude
stElev: station elevation (km) relative to sea level

Note that flags given to makeModel can change the interpretation of those fields with Lat, Lon, and Dep. See below for detail.

5.2.4 Starting Q Model

There are several options available for specifying the starting Q model. Options are 1-D homogeneous layers, 1-D gradient layers or full 3-D Q models. Both 1-D model variants are flat ASCII text files. The full 3-D model option uses a binary netCDF file format.

The 1-D homogenous layer file format contains a line for each layer and defines the top of that layer. The last line in the file contains the properties for the halfspace from that depth to infinite depth. The format is:

```
depth [Vp] Qp
```

where,

depth: Depth (km) to the top of the layer relative to sea level. Note that depths above sea level will be negative.
Vp: optional P-wave phase speed (km/s) for this layer
Qp: P-wave Q for this layer

The 1-D gradient layer file format is similar to that of the 1-D homogeneous case:

```
depth [Vp VpGrad] Qp QpGrad
```

where,

depth: Depth (km) to the top of the layer relative to sea level. Note that depths above sea level will be negative
Vp: optional P-wave phase speed (km/s) at the top of the layer.
VpGrad: if Vp is given, then this is required, otherwise it should not be given. This is the P-wave phase speed gradient (km/s/km) in depth. A positive gradient means that it increases with depth.
Qp: P-wave Q at the top of the layer
QpGrad: The P-wave Q gradient (1/km) in depth. A positive gradient means that it increases with depth.

Note that flags given to makeModel can change the interpretation of those fields with Depth. See below for detail.

The 3-D Q model input requires a netCDF format file (http://www.unidata.ucar.edu/software/netcdf/). Libraries and routines for reading and writing netCDF files in a variety of programming and scripting languages, including C, C++ and Matlab, are available through the above website. A netCDF file is a self-contained unit that defines dimensions, variables and attributes within a single file that is machine independent. Multiple variables may be contained within a single file and all the dimensions associated with those variables are contained within that same file.

Probably the easiest way to obtain a netCDF model is by using an existing model file output from makeModel or tstarTomog. One can always start off by using makeModel with a 1-D model to get an initial model file and then replace the output model by the desired 3-D model. The netCDF model requires several dimensions and variables to be defined. The required dimensions are:

NX: number of nodes in the X direction
NY: number of nodes in the Y direction
NZ: number of nodes in the Z direction
numCoord: 3 (always)

Along with these variables:

increments [numCoord]: float: array specifying [dx dy dz], i.e., the node spacings
x [NX]: float: array of the local x-axis. In Matlab format it will be: 0:dx:(NX-1)*dx
y [NY]: float: same as above, but for y
z [NZ]: float: same as above, but for z
xAxis [NX, numCoord]: float: 2-D array with the first column the same as 'x' above and the other two columns zeros

yAxis [NY, numCoord]: float: 2-D array with the second column the same as 'y' above and the rest zeros

zAxis [NZ, numCoord]: float: 2-D array with the third column the same as 'z' above and the rest zeros

qp [NZ, NY, NX]: float: 3-D array of Qp. The z-axis point positive down (depth), y-axis positive north, and x-axis positive east. The first data point is at the top, southwest corner of the model. x increases fastest, then y, finally z. See 'origin' below for the absolute reference frame.

qs [NZ, NY, NZ]: float: optional 3-D array of Qs. Otherwise the same as 'qp'

origin [numCoord]: double: array specifying [lat0 lon0 z0]. This gives the absolute location of the top, southwest corner of the model. z0 is depth (km) relative to sea level, so negative z0 is above sea level.

5.2.5 Seismic Velocity Models

A Vp and, if Qs is requested, Vs model is required. The Vp model can be given as part of the 1-D Q models as described above. A 3-D Vp (and Vs) model can also be given and, like for the 3-D Q models, require a netCDF file format. This can most easily be done by using an existing seismic velocity model as output from tomog (seismic velocity inversion algorithm; Preston, 2003). The netCDF file format for Vp (and Vs) is very similar to that of the 3-D Q model as described above. The only difference is that the variable(s) below must be included:

vp [NZ, NY, NX]: float: 3-D array of Vp (km/s). This array is laid out the same as 'qp' above

vs [NZ, NY, NX]: float: optional 3-D array of Vs (km/s). Same as above.

These variables can be included in the same file as 'qp' (and 'qs') or they can be given in a different file. Note that if they are given in a different file that all of the dimensions listed for the 3-D Q model along with all the variables (except 'qp' and 'qs') must be in this file. Also note, that the model type given for Q can be different than that for Vp, meaning that a 1-D model can be used for Q, while a 3-D model is used for Vp, or vice versa.

5.3 makeModel

The program makeModel reads in all input files and flags and produces a single output netCDF model file, which can be read by tstarTomog. These are the flags currently supported by makeModel:

-c *cdfFile*: model to create (req)
-s *staTabFile*: file containing station table
-z *zone*: utm zone number for model
-f[a|d] *fixfile*: fixed arrival table (a), or differential time table (d) (at least one is req)
-l *dx dy dz*: x, y, z increments (required if -v3 not given, otherwise ignored)
-i *dh*: travel time grid increment [smallest of *dx*, *dy*, *dz*]
-la *latmin latmax*: limits of lat (or Y) for model [determined by stations and events]
-lo *lonmin lonmax*: limits of lon (or X) for model [determined by stations and events]

-m *model.cdf*: read data from file *model.cdf*. The following options apply to this model until another -m
-mo: read observations from this file [no]
-mT: read topography from this file [no]
-mw: extra weight applied to all the picks in this file (multiplied) [1.0]
-mx: do NOT read picks from this file. Default is to read picks from the file
-n *minPh*: minimum number of amp picks for an event [1]
-nw : only count alive picks in number of picks to compare against above [all picks]
-o *obsFile*: observation file with '`lon lat depth qp weight`' [none]
-or *lon0 lat0 z0*: manually set the coordinate origin (or *x0,y0,z0*)(SW top of model)
-oz[s] *qp weight*: set observation on bottom-most layer of model to *qp* with *weight* [none] [s] does the same except for S velocities
-oz0 *qp weight*: set observation on top-most layer of the model to *qp* with *weight* [none]
-pm: assume input units on the most recently specified pick file is meters
-po *lon0 lat0*: use this as the origin lon/lat for the most recently specified pick file. This automatically assumes that the file is in X-Y coordinates.
-ps *psrat*: set the Qp to Qs ratio. If this flag is set and no Qs-model is given, then this will be used to make an Qs-model from the Qp-model. If an Qs-model is given the parameter is ignored unless the -sv flag is also given
-pz: assume that arrTab and diff arrTab events Z is elevation instead of default depth
-sv: force a conversion from the Qp-model to Qs-model even if an Qs-model is given
-tg *topoGridFileName*: gridFloat format topographic data from USGS seamless server [none]
One of the following is required (both a Q and Vp model are required):
-v[1] *1dmodFile*: file containing 1d model params with '`depth [vp] qp`' (km/s)
-vg *1dgradmodfile*: file containing 1d gradient params with '`depth [vp gradient] qp gradient`' (km/s/km)
-v3v *cdfFile*: cdf file containing a 3d velocity model to use. *nx*, *ny*, *nz*, *dx*, *dy*, *dz* determined from file
-v3q *cdfFile*: cdf file containing a 3d Q model to use. *nx*, *ny*, *nz*, *dx*, *dy*, *dz* determined from file
-vL: use the local data to determine *nx*, *ny*, *nz* and *origin* even when -v3* is given.
-vz: assume files for -v[1] and -vg flags are 'elevation ...'. Must be given before -v* flags.
-xs *sta1 sta2 ... staN*: list of station names to NOT use
-xp *pf1 pf2 ... pfN*: list of pickFile names to NOT use
-xP: do NOT use P picks
-xS: do NOT use S picks
-xy: assume all input following this option is in X-Y coordinates, not lat/lon. This option affects the -s, -e[a|d], -f[a|d], -la, -lo, -or, -o, -x[a|d] flags.
-uam: assume all units on the command line or in all files are in meters. Applies to all options and files following -uam. Equivalent to -um -up -uv
-uaz: assume all input is in elevation instead of depth both in command line options and files. Applies to all options and files following -uaz. Equivalent to -uz -pz -vz

-um: assume all input option units for distance are in meters instead of kilometers. Options that are affected: -l, -i, -la, -lo, -or, -o, -zm, -zn, -zm. Applies to all options following the -um flag.
-uk: assume all input option units for distance are in kilometers (default). Options that are affected: -l, -i, -la, -lo, -or, -o, -zm, -zn, -zm. Applies to all options following the -uk flag.
-up: assume all arrTab, diff arrTab, obs and station files following use meters instead of km. This can be used in lieu of -pm flags following each file. Note that the -s file is processed at the end.
-uv: assume velocity models given by -v[1] or -vg have depth in meters instead of default km. Velocity is assumed to be m/s and the velocity gradient is m/s/m. Must be given before -v flags.
-uz: assume -zm and/or -zn flag specify ELEVATION instead of depth. Must be given before -zm, -zn
Note that -um and -uk can be utilized multiple times in the command line to specify the units in one or more of the listed flags until another -um or -uk is encountered. Default is -uk.
-zm *maxZ*: maximum DEPTH of model (below sea level) [determined by events]
-zn *[minZ]*: do not use the *zmin* from the model (-v3 option) but calculate it anew. If *minZ* is provided then use this as the minimum DEPTH of the model
-z0 *baseElev*: use this elevation as the zero depth reference [0.0]

In the above listing, those flags that are required in all runs say '(req)' at the end of the flag description. The default value (if any) is provided in brackets at the end of the description. As mentioned above when describing the file formats, several of these flags can change the interpretation of the fields of the input text files. The file format descriptions give the default interpretation. The flags above can be used to change the interpretation of all depth fields to be that of elevation so that the sign of everything is the same. In addition, instead of latitudes and longitudes being the absolute location units, XY coordinates (such as UTM) can be given instead. When XY is specified, the 'Lat' fields become interpreted as 'X', while 'Lon' fields become 'Y'. Also, instead of km and km/s being the default units, all lengths can be interpreted as meters and velocities as m/s.

As an example, here is a call to makeModel:

```
makeModel -c tstarTest.cdf -s stationTab.txt -fd tstarTest.arrTab
-l .001 .001 .001 -i .001 -v q1d.txt -v3v vp3dModel.cdf -vL -z0 0
-zn -1.5 -zm -1.4 -oz 50 1
```

In this call (all on a single line), we will be creating a model file called tstarTest.cdf using a differential t* arrTab file tstarTest.arrTab with a station table stationTab.txt. We want the node spacing to be .001 km (1 m) in all three dimensions and the increment for the travel time grid to be .001 km (1 m). Use a 1-D homogeneous layered Q model from file q1d.txt and a fully 3-D Vp model from file vp3dModel.cdf. We want the locations of the sources and receivers of the data to determine the extents of the model instead of using the 3-D Vp model extents. Clearly

specify we want the zero depth reference to be sea level (generally recommended to avoid confusing the meaning of other input depths/elevations). We want the top of the model to be at a depth of -1.5 km (1500 m elevation) and the bottom of the model to be at a depth of -1.4 km (1400 m elevation) for a total model depth extent of 100 m. We also specify that we want the Q at the bottom of the model to be constrained to a value of 50 with a weight of 1.0.

If makeModel fails or runs into a problem, it will often give an explanation plus print a list of the flags. Common mistakes are swapping latitudes and longitudes and confusing depths and elevation fields. Also, make sure that the arrTab files have the proper number of columns and that all lines are complete.

6 RUNNING THE MODEL

Once a starting model is developed by makeModel, one is ready to perform an inversion with tstarTomog. Here we will describe only the most common flags used to control its operation. The appendix contains the full list of options.

modelName.cdf: The starting model name is given immediately following the executable name without any flags. This model file can either be one produced by makeModel or it can be the output from a previous tstarTomog run. (required)

-m *modelOut*: *modelOut* gives the base model name for this run. All output files associated with this run will begin with this prefix and will be placed, by default, into a directory called `data` in the current working directory that should already exist before tstarTomog is executed. Output model files will use this prefix followed by a two digit number starting from 00 and incrementing each major iteration, followed by a .cdf suffix. The zeroth iteration (output from starting model forward problem) is model 00. The model following the first inversion step is model 01. (required)

-n *N*: This flag specifies how many threads it should use. You get a nearly perfect speedup for the forward portion of the problem, but only a log(*N*) speedup for the CGLS inversion step. (required)

-f: Specifies that the 00 model should be written out. This will give you the starting predicted t* based on the starting model. Default is to only write out major iterations starting with the 01 model.

-fr *[vpModel.cdf]*: This flag indicates where it should read the Vp (and Vs) model from. *vpModel.cdf* is optional. If not provided, then the variable ‘vp’ (and ‘vs’) is expected to be found in *modelName.cdf*. This should be true of any starting model produced by makeModel. If *vpModel.cdf* is provided then it will read variable ‘vp’ (and ‘vs’) from that file instead. Only starting models produced by makeModel will contain both Q and Vp information. If you are using the output from one run of tstarTomog as *modelName.cdf* for a

new run, then 'vp' will not be present in that model. In this case, simply use the same starting model file you used to run *modelName.cdf* for *vpModel.cdf*. (required)

-dc: Perform distance corrections internally. This will use the length of the ray paths to determine these corrections. Do not use this flag if you already corrected the amplitudes for distance. Also, this flag is probably not that useful in its current form for absolute amplitudes, since the implicit assumption here is that the reference source-receiver distance is 1 km. For absolute data, you should distance correct the data yourself. I may update this feature in the future to allow one to specify the reference distance instead of the implicit assumption.

-r γ : γ gives the scalar weight of the entire regularization portion of the algorithm (see above). This weight is directly applied to the Laplacian smoothing portion of regularization, but it is a multiplicative factor for the station and source correction regularization. Larger weights increase model smoothness. (required)

-cs α : α gives the scalar weight of the receiver correction portion of the algorithm (see above). The actual weight applied is $\alpha\lambda$. Larger values of this combined weight force station corrections to be smaller.

-cxf β : β gives the scalar weight of the source correction portion of the algorithm (see above). The actual weight applied is $\beta\lambda$. Larger values of this combined weight force smaller source terms. This flag will convert "fixed" (unchangeable) sources into sources that allow source corrections.

-rC [1, 2]: Select the type of Laplacian regularization. 1 is the relative Q based regularization; 2 is Q based regularization. See above for a more in depth discussion. Do not include this option if you want the default 1/Q based regularization.

-ra ζ : ζ gives the scalar anisotropic vertical smoothing weight. Values less than 1.0 make the vertical direction rougher than XY.

-i *Dh*: This provides the travel time grid node spacing. This does not have to match what was given in makeModel. Typically this should be equal to half the minimum inversion grid node spacing (-l flag to makeModel). It is also acceptable to have this equal to the minimum inversion grid node spacing in many cases. Smaller values of this increase runtime for the forward modeling portion by a factor of about 8 for each halving of *Dh*, but it also increases ray path accuracy.

-mp *qpLimFile.txt*: This flag sets up bound parameter inversion. The file is a flat ASCII text file which gives the minimum and maximum allowed values of Q as a function of depth. The format for each line is

```
depth minQ maxQ
```

depth: depth to the top of the zone in km relative to sea level, so negative values are above sea level.
minQ: minimum Q value allowed in this zone
maxQ: maximum Q value allowed in this zone

A given line applies the bounds to all nodes that fall between its depth and the depth given on the next line. The final line applies its bounds to all nodes deeper than it.

An example command line for tstarTomog is:

```
tstarTomog startingModel.cdf -m modelA -n 4 -dc -f -fr -cs 1 -cxf 1 -r
1e-5 -rC 1 -i .0005 -mp qpLim.txt -ra .25
```

This call will read the data and starting model from startingModel.cdf and use the prefix "modelA" for all model output. We requested 4 threads and will be using internal distance corrections. Write out the zeroth iteration. The seismic velocity model is found in startingModel.cdf. We request station and source corrections with a weight of 1.0. The overall regularization weight is set to 1e-5 and we request that relative Q be used as the basis for smoothing regularization. The travel time grid node spacing is set to 0.0005 km. A Q model bounds file is provided. Finally, a vertical anisotropic smoothing weight of 0.25 is applied.

This model will run until the default convergence criteria are all met. Output will be placed in the default "data" directory with a new output file being written at the end of each major iteration. Diagnostic output is written to stderr. This output shows progress, RMS data misfits, smoothness, RMS of station and source correction terms, and much more. At the end of each major iteration, the complete variance history of all the iterations is summarized so that one can glean information on how the model is converging. Also, the convergence criteria and current values of these criteria for the current iteration are provided. Usually the final model iteration directly before program exit is the best model; however, one should scan the variance history to ensure that this is truly the case. tstarTomog will exit if the model is worse than the previous iteration.

7. PROGRAM OUTPUT

Output from tstarTomog consists of a series of netCDF files, one for each major iteration. As mentioned at the end of the last section, usually the best model is the final output before exit, but this should be checked using tstarTomog's diagnostic output.

The format for the netCDF output from tstarTomog is virtually identical to that of makeModel except that tstarTomog does not write the seismic velocity variable(s) and tstarTomog writes out raypath coverage data to the output file. This section will give the dimensions and variables

defined for both program outputs. C/C++ and Matlab, along with most other common programming languages, provide netCDF support so that one can read and display output.

Dimensions:
numCoord: 3 (always)
NX: number of nodes in the X dimension
NY: number of nodes in the Y dimension
NZ: number of nodes in the Z dimension
numObservations: number Q points with constraints (only if constraints are provided)
numStations: number of unique stations
stationNameLen: 10 (always). This is maximum length for a station name.
numPickFiles: number of unique events
pickFileNameLen: 512 (always). This is maximum length for an event name.
numPicks: number of t* observations
numDtPicks: number of differential t* observations (only if there is at least 1)

Variables:
increments [numCoord]: float: array specifying [dx dy dz], i.e., the node spacings
x [NX]: float: array of the local x-axis. In Matlab format it will be: 0:dx:(NX-1)*dx
y [NY]: float: same as above, but for y
z [NZ]: float: same as above, but for z
xAxis [NX, numCoord]: float: 2-D array with the first column the same as 'x' above and the other two columns zeros
yAxis [NY, numCoord]: float: 2-D array with the second column the same as 'y' above and the rest zeros
zAxis [NZ, numCoord]: float: 2-D array with the third column the same as 'z' above and the rest zeros
qp [NZ, NY, NX]: float: 3-D array of Qp. The z-axis point positive down (depth), y-axis positive north, and x-axis positive east. The first data point is at the top, southwest corner of the model. x increases fastest, then y, finally z. See 'origin' below for the absolute reference frame.
qs [NZ, NY, NZ]: float: optional 3-D array of Qs. Otherwise the same as 'qp'
qpSensitivity [NZ, NY, NX]: float: 3-D array of weighted P-wave ray travel time per node. Each element is simply the sum of the column of the A matrix corresponding to that point. This gives an idea of the sensitivity of the model as a function of 3-D space. Zeros mean that no ray "hit" that node. makeModel does not output this variable.
qsSensitivity [NZ, NY, NX]: float: 3-D array of weighted S-wave ray travel time per node. This exactly corresponds to qpSensitivity. This will only be output if qs is given and there are S-wave t* observations.
vp [NZ, NY, NX]: float: 3-D array of Vp (km/s). This array is laid out the same as 'qp' above. tstarTomog does not output this variable.
vs [NZ, NY, NX]: float: optional 3-D array of Vs (km/s). Same as vp.

origin [numCoord]: double: array specifying [lat0 lon0 z0]. This gives the absolute location of the top, southwest corner of the model. z0 is depth (km) relative to sea level, so negative z0 is above sea level.

observationsX [numObservations]: float: array specifying the local model X coordinates for all of the Q constraint points. This will only be present if numObservations exists.

observationsY [numObservations]: float: array specifying the local model Y coordinates for all of the Q constraint points. This will only be present if numObservations exists.

observationsZ [numObservations]: float: array specifying the local model Z coordinates for all of the Q constraint points. This will only be present if numObservations exists.

observationsVal [numObservations]: float: array specifying the Q constraint values for all of the Q constraint points. This will only be present if numObservations exists.

observationsWeight [numObservations]: float: array specifying the weights for all of the Q constraint points. This will only be present if numObservations exists.

observationsType [numObservations]: int: array specifying the type (0 for P-waves; 1 for S-waves) for all of the Q constraint points. This will only be present if numObservations exists.

stationName [numStations, stationNameLen]: char: 2-D array of the unique station names.

stationX [numStations]: float: array specifying the local model X coordinates for the stations.

stationY [numStations]: float: array specifying the local model Y coordinates for the stations.

stationZ [numStations]: float: array specifying the local model Z coordinates for the stations.

stationTp [numStations]: float: array specifying the P-wave station correction terms for the stations. This will only be present if station corrections are requested.

stationTs [numStations]: float: array specifying the S-wave station correction terms for the stations. This will only be present if station corrections are requested.

pickFileNames [numPickFiles, pickFileNameLen]: char: 2-D array of the unique event names.

pickFileX [numPickFiles]: float: array specifying the local model X coordinates for the events.

pickFileY [numPickFiles]: float: array specifying the local model Y coordinates for the events.

pickFileZ [numPickFiles]: float: array specifying the local model Z coordinates for the events.

pickFileT [numPickFiles]: float: array specifying the P-wave source correction terms for the events.

pickFileTs [numPickFiles]: float: array specifying the S-wave source correction terms for the events.

pickFileX0 [numPickFiles]: float: deprecated.

pickFileY0 [numPickFiles]: float: deprecated.

pickFileZ0 [numPickFiles]: float: deprecated.

pickFileT0 [numPickFiles]: float: deprecated.

pickFileRegWeight [numPickFiles]: float: deprecated.

pickFileRegZ [numPickFiles]: float: deprecated.

pickFileRegT [numPickFiles]: float: deprecated.

pickFileType [numPickFiles]: int: array specifying the type of the event. This is primarily for internal use, but gives whether the event is from an earthquake (0), fixed source (1) or explosion (2). Explosive types simply allow source corrections to be applied for otherwise fixed sources.

pickFileStartPickIndex [numPickFiles]: int: array that gives the starting index number (starting at 0) for each event of data in arrays of length numPicks.

pickFileStopPickIndex [numPickFiles]: int: array that gives the last index number for each event of data on arrays of length numPicks. This information is basically just for ease. This is because pickFileStopPickIndex[i] = pickFileStartPickIndex[i+1] except for the last entry in which pickFileStopPickIndex[last] = numPicks. In Matlab, for example, to get all the observed t* values for event i, one would use pickObsTStar(pickFileStartPickIndex(i) +1:pickFileStopPickIndex(i)).

pickName [numPicks, stationNameLen]: char: array of the station names associated with every t* observation.

pickObsTStar [numPicks]: float: array of t* observations.

pickCalcTStar [numPicks]: float: array of predicted t*. Note that these will have entries of -1000 for picks whose rays went out of bounds. These are not updated if a pick has been thrown out.

pickFreq [numPicks]: float: frequencies of the t* observations.

pickWeight [numPicks]: float: weights applied to the data. Typically these are 1.0 unless a ray is out of bounds or data have been downweighted or thrown out due to too large an error. Downweighted, but still used picks will have weights between 0 and 1. Weights of 0 or negative have been thrown out of the inversion and are not used.

pickType [numPicks]: int: array that indicates the type of t* observation. This can be P-wave absolute (0), S-wave absolute (1), P-wave differential (4), S-wave differential (5).

pickETime [numPicks]: float: array of the estimated standard error of the data. These are simply those input into makeModel when the data was first built.

dtPickOtherEventIndex [numDtPicks]: int: array of the indices into arrays of length numPickFiles for the second event in a differential pair. This will not be present if there are no differential picks in the data set.

dtPickOtherStatName [numDtPicks, stationNameLen]: char: array of the names for the second station in a differential pair. This will not be present if there are no differential picks in the data set.

Attributes:

isFlat: int: This indicates whether the model assumes a flat earth geometry (1) or if it includes spherical earth corrections (0).

history: char: This gives the command line call that produced this file.

The dimensions of each variable is given in brackets after the name of the variable, followed by the variable type (float, double, char, int). The variables are C-style in that all variables that refer to indices are 0 based. Also, the layout in memory is such that the final dimension of a variable is the one that varies the fastest.

8. RESOLUTION AND ERROR ANALYSIS

Direct covariance analysis would be ideal but this involves the computation of $\mathbf{A}^T\mathbf{A}$, which is, in general, not sparse. This can result in a very large filled matrix, which is extremely

computationally burdensome for even relatively small realistic problems and intractable for large problems. As such, we do not compute the direct covariance of the problem, but instead, use model resolution and statistical jackknife tests to estimate the model error.

8.1 Resolution Tests

A resolution test involves the imposition of a known pattern to the model and the attempted recovery of that pattern. A common resolution test used in tomography is the checkerboard resolution test. In this test, the final 3-D model is altered by multiplication of the 3-D Q values by a regular pattern of 1+r and 1-r blocks that in 2-D looks like a checkerboard. The quantity r is the amplitude or peak fractional change to the final model, with values of 0.05 and 0.10 typical. This perturbed model is then used to calculate t* values at all the data points. These t* values become the "observed data" of the resolution test. Gaussian distributed errors also can be added to these t* values to more realistically represent actual error and variation in the data. The inversion then proceeds as usual until it converges. The output of this converged model is called the resolution model. In order to interpret the resolution of the model, one displays the ratio of the resolution model to the original model. In a perfectly resolved model, this will show the same regular pattern of 1+r and 1-r blocks that one input. In realistic scenarios, there will be smearing across blocks, blocks with returned amplitudes either greater or smaller than the input amplitudes, and regions with absolutely no recovery.

tstarTomog has built-in options to create checkerboard test runs. Besides the block pattern mentioned above, one can also choose a more smoothly varying checkerboard pattern based on 3-D sine functions. Both Qp and Qs can be tested. Errors can be added to the resolution t* values. These options are controlled by several different -T flag variants. See Appendix A for a list of these options and how they control the resolution tests. In order to test the resolution of a given model, it is best to use as many of the same flags in the resolution tests as were used to make that model. Especially important is the overall regularization weight (-r flag) and vertical anisotropy regularization weight (-ra flag). You may exclude external Q model (*a prior*) constraints from resolution tests (-rxn flag) or force the external constraint values to match those of the perturbed model (-Tx flag), if desired.

8.2 Error Analysis

Model parameter standard error is estimated via the statistical jackknife test (e.g., Wonnacott and Wonnacott, 1985). For this test, a certain fraction (e.g., 10%) of the data is removed from the data set and a full inversion is done on this subset. Another nine models are then run, each with a different 10% of the data removed. This gives 10 models. These 10 models can then be analyzed to estimate the standard errors. tstarTomog has limited capability for handling jackknife tests directly. One flag, -j, basically turns off certain flags that would normally throw out events that have too few data, etc. There are no flags in tstarTomog that will actually make the 10 models with 10% of the data removed from each. These models must be made outside of tstarTomog and each run independently. These 10 runs then need to analyzed by a program outside of tstarTomog to actually give standard error estimates. Please see standard statistical papers and books for the best jackknife test equations for your particular problem. From

experience, I find that the equations given in Wonnacott and Wonnacott (1985) give reasonable looking distributions of high and low standard errors, but that the values appear much too small.

9. REFERENCES

Bierlaire, M., P.L. Thoint, and D. Tuyttens, On iterative algorithms for linear least squares problems with bound constraints, *Lin. Alg. and its Appl.*, 143, 111-143, 1991.

Hole, J. and B. Zelt, 3-D finite-difference reflection traveltimes, *Geoph. J. Int.*, 121 (2), 427-434, 1995.

Lay, T. and T.C. Wallace, *Modern Global Seismology*, Academic Press, San Diego, 521 pp., 1995.

Paige, C. and M. Saunders, LSQR: An algorithm for sparse linear equations and sparse least squares, *ACM Trans. on Math. Soft.*, 8, 43-71, 1982.

Preston, L.A., *Simultaneous Inversion of 3D Velocity Structure, Hypocenter Locations, and Reflector Geometry in Cascadia*, Ph.D. Thesis, University of Washington Seattle, Seattle, WA, 2003.

Vidale, J., Finite-difference calculation of traveltimes in three dimensions, *Geophysics*, 55(5), 521-526, 1990.

Wonnacott, R.J. and T.H. Wonnacott, *Instrductory Statistics*, John Wiley and Sons, New York, 649 pp., 1985.

APPENDIX A: ALL PROGRAM OPTIONS

This appendix lists all the options available for tstarTomog. The most common options are given above in the "Running tstarTomog" section (Section 6). Some of these options here could be useful in some circumstances, but you should use many of them with caution.

argument: model name--this must be a cdf file as described above. Includes a Qp and Qs (optional) model and all the data to be used in the inversion. It may also include Vp (and Vs)

-v [#]: Verbose option, followed by optional digit sets value of Verbose global to that value. This allows varying levels of verbosity. Default value of Verbose is 1.
-n #: use multi-threaded OpenMP version with # threads.
-m *name*: Sets global ModelNameRoot. Each iteration is a derivation of this.
-d : decimate the input model by a factor of two. It will cover at least the same volume as the original

-di *xyz* : interpolate the model by a factor of two. This effectively un-does the effect of -d. To get the same model size back as originally, you need to provide a three digit number, *xyz*, that signifies whether the length of x, y and z are even or odd. a '1' means make it odd, a '2' means even, so -di 122 would make *nx*'s length odd and *ny* and *nz* lengths even.

-dT *cdfFile*: read topography from this model file. Useful with the -di option to get higher res topo into the model

-i #: This determines the grid spacing for the travel time calculation grid. <2.0>.

-f: Write out zeroth iteration. Useful if you want calculated times for a different model than is already in the file or just want calculated times for a starting model.

-fl 1 or 0: use/convert to a flat earth (1) or use/convert to a spherical earth approx (0). Only z is affected by this approximation and will be valid out to several degrees distance. The output model will have its global attribute 'isFlat' set to the value of -fl. <1>

-j [# [#]]: doing jackknife tests. Turns off testing for min number of live picks and all pick tossing (exceeding standard devs, etc.) and does not allow bound inversions. If a number is given, the slownesses will be multiplied by this factor before inversion begins. If solving for Qp/Qs, Qp/Qs will be mult by #. If 2 numbers are given then the first applies to P and the 2nd to S or Qp/Qs. <1.0>

-bi #: number of allowed major iterations with an increase is misfit before bailing <1>

-cx[f] #: use this # for explosion time adjustment regularization. If 'f' is also given, then all fixed type are converted to explosions. <10000>

-cs #: use this # for station correction regularization. If this option is not given, then no station corrections will be solved for.

-mm *cdffile*: restart where you left off. The model name you give above should be the final model of the previous iteration before CGLS inversion. The model given on this line should be the one after the CGLS inversion. It compares the sizes and does what it normally would right after exiting from the CGLS inversion.

-mp *file*: use this file to construct min and max allowable Qp values in the model as a function of depth. Format is "depth(km) q_min q_max". All depths below the last given will use those limits. All depths above the min depth will be given the limits for the shallowest layer. A "-1" in either the min or max values for a depth will take the corresp default values < 1.5 8.0>

-ms *file*: use this file to constrain min and max allowable Qs values in the model. If Qp/Qs is used or Qp/Qs reg is requested, the min and max values will apply to the Qp/Qs model. Everything else is just like -mp. Defaults are:< .01 5 > for Qs and < 1.6 2.5 > for Qp/Qs.

-pr: reset all the negative pickweights to positive and any 0 weights to 1.

-pi: reset all pickweights<0 each iteration. This makes it calculate rays for all picks regardless of whether they have been thrown out in a previous iteration.

-pw *min max*: sets the min and max data error allowed. Any pick with an edata > max will have its pickweight adjusted such that pw/edata = 1/max, etc. Does not affect differential picks.

-pwr: reset all pick weights > 0 to 1.0

-e: The -e? family of flags control various aspects of how events are dealt with

-es #: This sets the initial number of elements allocated in the A-matrix for each ray. If this is too low then element and column arrays will need to be reallocated, if too big A will be very inefficient in its space usage. Default is generally fine. <200>.

-en #: minimum number of live picks to keep an event <1>.

-ev *n v*: keep all picks regardless of residual for *n* full iterations and then throw out picks with weighted residuals > *v* times the standard deviation for that event

-evr *n v*: keep all picks regardless of residual for *n* full iterations and then throw out picks with residuals > *v* seconds

-evv *n v*: keep all events regardless of variance for *n* full iterations and then throw out events with weighted standard deviations > *v* (unitless)

-r: the -r? family of flags control various aspects of the regularization

-r[w] #:Sets the overall regularization weight. All other regularization weights are multiplied by this number. <10000>

-rv *regW regWMult*: Use variable regularization weighting. *regW* is the starting regularization weight, *regWMult* is between 0 and 1 and is applied to *regW* after each full iteration until the reg weight given by the -r flag, ie. the -r flag gives the minimum reg weight

-rl #: Use distance weighting. The weight given an observation is multiplied by the length of the ray times the number given. Note that this weight can never weight a pick > 1, so effectively any ray less than the inverse of the # in length gets a weight of 1. To change the overall weighting use -rlw in addition to this one.

-rlo: Only use distance weighting, i.e., the estimated pick error times are not used, only the distance weighting weights picks. <FALSE>

-ra #: Sets the vertical regularization anisotropy. Numbers less than 1 allow more variability in the z-dir. <0.1>

-ray #: Sets regularization Y anisotropy. Numbers >1 allow smoother models in the Y dimension handy for quasi 2D models. <1.0>

-rz: Toggles the value of the ZeroEdge. This effects the behavior of the regularization at the model edges. If ZeroEdge is true the perturbations are constrained to be zero at the edge. If false the slope of the perturbation is set to zero for creeping, or the slope of the model is set to zero for non-creeping solutions. <FALSE>.

-rc: Changes to creeping regularization by setting the Creep to TRUE. This means the smoothness of individual perturbations is constrained to be zero. The converse means the model is smoothed by setting b=-Lx. <FALSE>

-rx *file*: Use the following file as external Q constraints. The file is of the form:

```
x y z qp weight
```

-rxn: Do NOT use any external constraints in the inversion. Any present will still be written to the model output files, but will not be used. This may be helpful for testing the effects of the constraints and for resolution (checkerboard) tests. <use external constraints>

-rf *factor*: relative amount that the Qs/QpQs model is smoothed relative to Qp. Default is 1.0/ sqrt(3). Negative values will seek a flat solution.

-rb: use QpQs regularization for Qs. This causes the program to solve for QpQs instead of Qs values. Qs will be output. <use Qs directly>

-rp *file*: Prints the ray path information to the filename following this flag. File is in the form "station source" followed by lines of x y z. Will quit after one iteration.
-rS[ps] *weight*: use small perturbation regularization for Qp ('p' or none) or for Qs (or QpQs) ('s')
-rC #: # is either 1 or 2. If 1, then the regularization will be weighted such that smoothness will be measured in terms of normalized local Q so the smoothness is related to $(Q_0-Q_1)/Q_0$. This means that, for example, any place that has a change of say 5% in Q will have the same smoothness. If # is 2, then the regularization will be weighted such that smoothness will be measured in terms of Q, i.e., proportional to Q_0-Q_1. This means that any place where the Q difference is, for example, 10, the smoothness would be the same. The default behavior is that the smoothness will be measured in terms of $1/Q_0-1/Q_1$ (i.e., loss factors), so that, for example, any 0.1 difference has the same smoothness. <use 1/Q for smoothness>

-wd *dirname*: Set the location where most data files are placed. This directory must already exist. <"data">.

-s: The -s? family of flags controls the solver (Both the linear and the non-linear portions)
-sn #: Sets the value of the node error variable. This is the RMS of the perturbation vector for the loss factor model (1/Q). Enter this number as the % mean error in Q per node allowed for convergence. <10> (%)
-sN #: Set the value of the max node fractional change between iterations. <0.05> (5%)
-sxr #: Sets the value of the maximum fractional change in RMS allowed for convergence. <0.05>
-sx #: Sets the maximum node change allowed for convergence in loss factor (1/Q) units. <0.00175>
-sxc #: Sets maximum perturbation norm allowed for convergence <10>
-sp *lim1 lim2*: Sets downweighting for picks whose abs weighted error exceed certain limits. For picks whose weighted error are less than *lim1* standard deviations then use standard errors as weights. If it falls between *lim1* and *lim2* standard deviations, then use a cosine taper to downweight picks. Outside of *lim2* are given zero weight but not removed so that they can come back.
-sg #: Starting gamma value. Useful if want to start in the middle of a gamma run so you don't have to do the whole iteration. This along with the -ss option allows you to start in the middle of any iteration and thus can use times calculated on a previous run.
-ss #: Starting iteration value. Can be used with -sg. See -sg.
-si #: Sets the maximum number of non-linear iterations. <25>.

-T *testFile*: create a resolution test file named "testFile.cdf"
-Tv *vAmp [x0] xL [y0] yL [z0] zL*: A Q model with sine checker-board pattern centered about vp with fractional amplitude *vAmp* and wavelengths *xL*, *yL*, *zL*. *x0*, *y0*, and *z0* are optional (but if one is given, then all the others must be also) and shift the checkerboard pattern by X km in the direction given. -T must be given as well. If *xL*, *yL*, or *zL* is <=0, then there will be no functional dependence in that direction.

-Tb *vAmp [x0] xL [y0] yL [z0] zL*: A Q model with box car checker-board pattern centered about vp with fractional amplitude *vAmp* and wavelengths *xL*, *yL*, *zL*. *x0*, *y0*, and *z0* are optional (but if one is given, then all the others must be also) and shift the checkerboard pattern by X km in the direction given. -T must be given as well. If *xL*, *yL*, or *zL* is <=0, then there will be no functional dependence in that direction.

-Te[p|s] *Err*: Add gaussian distributed errors to calculated t*. The units are standard error units, so they scale the pick standard errors. So if a pick had a standard error of 0.05 s and you have the *Err* as 1.5, then the error used would be 0.075 s. This keeps the distribution of standard errors the same. An optional 's' may be added to -Te to supply S-wave error times. If these are not given, and S times are in the model, then the P errors will be used. 'p' may be added to the end, but if none is added, then 'p' is assumed. You must use -Te for both P and S errors if they exist, i.e. you can not mix -Te and -Tet.

-Ted[p|s] *Err*: Add gaussian distributed errors to differential t*. The units are standard error units. An optional 's' may be added to -Te to supply S wave error times. If these are not given, and S times are in the model, then the P errors will be used. 'p' may be added to the end, but if none is added, then 'p' is assumed. Note that if this option is not provided and there are differential data in the dataset, the default is 0.0 error even if -Te above is given. Note: you cannot mix -Te and -Tet flags within one run.

-Tet[p|s] *Err*: Add gaussian distributed errors to calculated t*. The units are actual error times in seconds, so .05 means the errors added to the times will have a gaussian distribution with a standard deviation of .05 seconds. An optional 's' may be added to -Tet to supply S-wave error times. If these are not given, and S times are in the model, then the P errors will be used. 'p' may be added to the end, but if none is added, then 'p' is assumed. You must use -Tet for both P and S errors if they exist, i.e. you can not mix -Te and -Tet.

-Tetd[p|s] *Err*: Add gaussian distributed errors to calculated differential t*. The units are actual units (s). An optional 's' may be added to -Te to supply S wave error times. If these are not given, and S times are in the model, then the P errors will be used. 'p' may be added to the end, but if none is added, then 'p' is assumed. Note that if this option is not provided and there are differential data in the dataset, the default is 0.0 error even if -Ted above is given. Note: you cannot mix -Te and -Tet flags within one run.

-Tx: Forces any external constraints to match the values of the perturbed model. This overwrites any external constraint values with values of the perturbed model.

-Tm: exit after the perturbed model times have been calculated instead of swapping velocities and inverting. With this option, you could then manually replace the Q model with whatever one you want instead of using the initial model

-Tp *cdfFile*: Use this model as the perturbed model (precludes -Tv, -Tb)

DISTRIBUTION

1	MS0750	Leiph Preston	6913 (electronic copy)
1	MS0750	Robert E. Abbott	6913 (electronic copy)
1	MS0750	Hunter Knox	6913 (electronic copy)
1	MS0899	Technical Library	9536 (electronic copy)


Sandia National Laboratories